International Journal of Intelligent Information Systems
2015; 4(2): 35-39
Published online February 14, 2015 (http://www.sciencepublishinggroup.com/j/ijiis)
doi: 10.11648/j.ijiis.20150402.11
ISSN: 2328-7675 (Print); ISSN: 2328-7683 (Online)

Local search heuristic for multiple knapsack problem

Balbal Samir[1], Laalaoui Yacine[2], Benyettou Mohamed[1]

[1]Computer science Department, USTOMB, Oran, Algeria
[2]IT Department, Taif University, Taif, Kingdom of Saudi Arabia

Email address:
belbelsamir@gmail.com (Balbal S.), y.laalaoui@tu.edu.sa (Laalaoui Y.), med_benyettou@yahoo.fr (Benyettou M.)

To cite this article:
Balbal Samir, Laalaoui Yacine, Benyettou Mohamed. Local Search Heuristic for Multiple Knapsack Problem. *International Journal of Intelligent Information Systems.* Vol. 4, No. 2, 2015, pp. 35-39. doi: 10.11648/j.ijiis.20150402.11

Abstract: In this paper we will present a heuristic method to solve the Multiple Knapsack Problem. The proposed method is an improvement of the IRT heuristic described in [2].the experimental study shows that our improvement leads some gain in time and solution quality against IRT, MTHM, Mulknap and ILOG CPLEX.

Keywords: Multiple Knapsack Problem, Local Search, Heuristic

1. Introduction

The Multiple Knapsack Problem (MKP) is a variant of the knapsack problem (KP) whose resolution is much more difficult, the fact that we have this problem in areas as different application than the economy, industry, transport, cargo loading and distributed computing, gives it a great practical interest [1].

Viewpoint Artificial Intelligence, the problem of Multiple Knapsack is strongly NP-complete. This means that the resolution of this problem cannot be done in polynomial time. In other words, an exact algorithm is required for optimal resolution.

The objective of this work is to improve the performance of a heuristic proposed by IRT Laalaoui [2], and solve the problem of multiple Knapsack in a way we approached using local search.

2. Presentation of the Multiple Knapsack Problem

The Multiple Knapsack Problem (MKP) is a generalization of the standard 0-1Knapsack Problem where instead of considering only one knapsack, one tries to fill m knapsacks of different capacities [3]. Consider a set $N = \{1\dots n\}$ of items to be loaded into m knapsacks of capacity c_i with $i \in \{1, \dots m\}$. Each item $j \in N$ is characterized by its weightw_j, and its profit $p_j$and its decision variable x_{ij} which is worth 1 if the item j is loaded into the knapsack i and 0 otherwise. It is then to find m disjoint subsets of N (where each subset corresponds to filling a knapsack) that maximize the total profit made by the sum of the selected items. The mathematical formulation of the problem MKP is as follows:

$$MKP: \begin{cases} \max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \\ s.c \ \sum_{j=1}^{n} w_j x_{ij} \le c_i \ , \ i \in \{1, \dots, m\} \\ \sum_{i=1}^{m} x_{ij} \le 1 \ , \ j \in \{1, \dots, n\} \\ x_{ij} \in \{0,1\}, i \in \{1, \dots, m\}, j \in \{1, \dots, n\}. \end{cases} \quad (1)$$

Where p_j , c_i and w_j are positive integers.

In order to avoid any trivial case, we make the following assumptions.

All items have a chance to be packed (at least in the largest knapsack):

$$\max_{j \in \{1..n\}} w_j \le \max_{i \in \{1,\dots,m\}} c_i \quad (2)$$

The smallest knapsack can be filled at least by the smallest item:

$$\min_{j \in \{1..n\}} w_j \le \min_{i \in \{1,\dots,m\}} c_i \quad 3)$$

There is no knapsack which can be filled with all items of N:

$$\sum_{j=1}^{n} w_j > \max_{i \in \{1,\dots,m\}} c_i \quad (4)$$

3. Resolution Method of MKP

The approaches proposed in the literature to solve the problems of the family of the backpack are either exact methods are heuristics. The exact methods are able to solve a problem to optimality but in exponential time [4]. Heuristic methods provide an approximate solution, good quality in reasonable periods of time [4]. Heuristics are either simple heuristic are meta-heuristics.

3.1. The Exact Method

The exact methods proposed in the literature to solve problem MKP are based on the Branch-and-Bound (B &B).

Ingargiola and Korsh [5] proposed a branch-and-bound algorithm which used a reduction procedure based on dominance relationships between pairs of items.

Hung and Fisk [6] proposed a method based Branch and Bound with depth-first strategy as a journey. The upper bounds are obtained using Lagrangian relaxation, with a decreasing scheduling capacity c_i.

The algorithm of Martello and Toth [7] improves proposed by Hung and Fisk with the calculation of upper bounds using surrogate relaxation and taking the minimum of the Lagrangian upper bounds and surrogate relaxation method.

Martello and Toth[8] proposed an algorithm (bound and bound) algorithm improves the Martello and Toth[7]a powerful base of B&B to solve the MKP. This algorithm, called MTM (Method Martello and Toth), applies heuristics Greedy, which involves solving a series of problems with *m* single Knapsack.

Pisinger[9] improved the algorithm MTM by incorporating an efficient algorithm for calculating higher and better reduction rules for determining the items that can be set to zero terminals and a method that attempts to reduce the ability of backpacks. This new algorithm is called Mulknap. Power Mulknap located in allocating 100000 items in one second. So Pisinger has succeeded with Mulknap resolve cases problems with very large (n = 100 000, m = 10) in a second. But at the same time it fails to resolve cases in smaller problem (n = 45, m = 15), when the ratio *n/m* is between 2 and 5 ($2 \leq n/m \leq 5$).

Fukunaga and Korf [10] proposed the bin-completion method is a technique based branch- and-bound. It uses the strategy depth first. Each node of the search tree represents a maximum possible allocation for a particular knapsack member.

A. Fukunaga [11] improved bin-completion method in the case of relatively large bodies (n = 100). But the ratio *n/m* is the major problem in all existing algorithms.

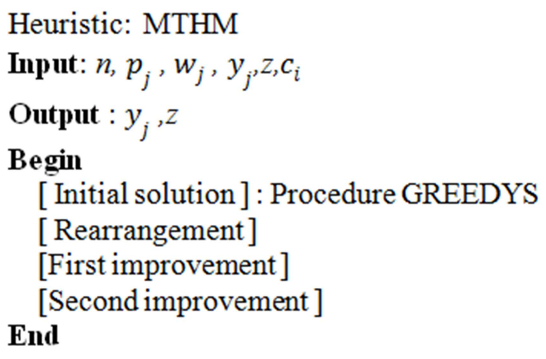
3.2. Existing Solvers

There are many solvers have been developed for solving the problem of the backpack. We distinguish between free software and commercial software. Commercial software often has superior performance to the free solvers. There are two principal existing business software is: The commercial solver IBM ILOG CPLEX and XPRESS-MP solver. There exist also two principal free software are: GLPK and Boob ++.

3.3. Heuristics

Heuristic methods have been proposed for the problem of multiple bag back in order to find good solutions within a reasonable time, heuristic MTHM, CRH and IRT are proposed to solve the problem MKP.

The heuristic (MTHM) of Martello and Toth [12] is a very efficient heuristic to solve the problem MKP It takes place in stages present in the following Figure.



```
Heuristic: MTHM
Input: $n, p_j, w_j, y_j, z, c_i$
Output : $y_j, z$
Begin
   [ Initial solution ] : Procedure GREEDYS
   [ Rearrangement ]
   [First improvement ]
   [Second improvement ]
End
```

Fig. 1. *Heuristic MTHM*

The heuristic RCH described by Lalami et al. in [13] is a heuristic with a polynomial time complexity for solving the MKP. Unfortunately, this heuristic resolve any problems that could be solved using optimality Mulknap i.e. instances of problems with a large *n/m* ratio, which is where the Mulknap gives the best results in less second. The authors fail to describe the interesting case of problems with a small ratio *n/m*.

In [2], Laalaoui proposed a heuristic to solve the problem completely dependent exchanges found in MTHM and also to increase the efficiency of the latter method (improved profit). This new heuristic integrates three simple heuristics (Replace-One-By-One, Replace-Two-By-One and Replace-One-By-Two) with MTHM by two different techniques: the first technique is simple (SRT) and the second iterative (IRT).

3.4. Metaheuristics Methods

Among the proposed literature to solve the problem MKP methods that uses genetic algorithms metaheuristic methods, methods are located: HGGA (Hybrid Grouping GA) [14], WCGA (Weighted Coding GA) [15], Ugga (Undominated Grouping GA) [16] and Representation-RSGA (Switching GA) [17].

4. Local Search Heuristic for MKP

Local Search is used by many metaheuristic. It is about making incremental improvements to the current solution through a basic transformation until no improvement is possible. The solution is called local optimum found with respect to the transformation used, as shown in Fig.2.

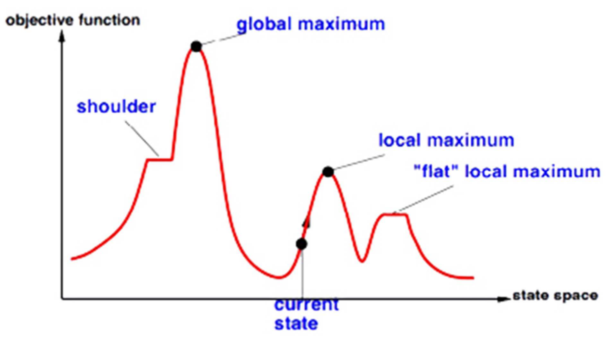


Fig. 2. *Local Search*

Technically, the local search consists of a series of transformations of the solution to improve it every time. The current solution S is replaced by a better solution S' ∈ N(S) in its vicinity. The process stops when it is no longer possible to find-improving solution in the vicinity of S, such that the algorithm written Fig. 3

```
Algorithm: Local Search
Input: S
Output: N
best ← true
**While** best = true **do**
best ← false
**for** (S' ∈ N(S)) **do**
**if** (S' is best of S)
S←S'
best ← true
**return**
```

Fig. 3. *Algorithm for Local Search*

Our proposal to solve the problem MKP with local search method is using the following steps:

Step 01: initial solution;

Step 02: Perturbation solution

Step 03: improve the solution;

Step 04: repeating the process a number of times.

4.1. Initial Solution

For the initial solution of this method we will use the IRT technique written by Y. Laaloui in [2].

4.2. Perturbation Solution

We know that one of the disadvantages of IRT and MTHM is the lack of randomness .This drawback severely limits the ability to better search space exploration.

In our new technical we introduce some randomness to the solution of step disturbance. The principle of perturbation solution is to randomly remove one item or several items of the solution as mentioned in the procedure Perturbation.

4.3. Improve the Solution

For the third step the procedures for exchanging items is applied (Replace-One-By-One, Replace-Two-By-One, Replace-One-By-Two) and the steps are repeated for a number of times. The figure (Fig .4) shows the general algorithm of the method of local search for MKP.

```
Algorithm Local Search Heuristic for MKP ;
Inputs : $n$ , $p_j$ ,$w_j$ , $y_j$ ,$z$, $i$ ,$\overline{c_i}$
Outputs : $y_j$ ,$z$
Begin
01.   MTHM
02.   Replace-Two-By-One
03.   Replace-One-By-Two
04.   Replace-One-By-One
05.   if( $prev\_z < z$ )
06.       $prev\_z = z$;
07.       goto step 02.
08.   if($tmpz == z$)
09.   $tmpz = z$;
10.   for ($j = 1; j \le d; j++$)
11.       $tmpx[j] = x[j]$;
12.   for ($j = 1; j \le b; j++$)
13.       $tmpcr[j] = cr[j]$;
14.   else
15.   $z = tmpz$;
16.   for ($j = 1; j \le d; j++$)
17.       $x[j] = tmpx[j]$;
18.   for ($j = 1; j \le b; j++$)
19.       $cr[j] = tmpcr[j]$;
20.   if($testmore <= nbretest$)
21.       Perturbation()
22.       $testmore = testmore + 1$
23.       goto step 02.
End.
```

Fig. 4. *Local Search Heuristic for MKP*

5. Experimental Results

To measure the effectiveness of our work, we have implemented in C programming language, this choice is justified by the speed of the language. And we used the system Lunix (Ubuntu) as a platform for development, since it is widely used in the academic community, and to use shell scripts. The technical Mulknap work is written in C1 .While the code of the implementation MTHM is written in FORTRAN2 and we converted to C using the f2c converter.

We used the optimization tool IBM ILOG CPLEX commercial solver version 12.2.5. All techniques are established in the same environment using the GCC compiler. All tests were performed on a 2.2 GHz Intel Core Duo 2 processor with 2GB of RAM. We have used A. Fukunaga's data-set which was used in [16][17]. This benchmark is a set of 12 problem instances, four instances in each one of the three types: strongly correlated, weakly correlated and multiple subset-sum. The number of knapsacks is 100; the number of items is 300 in each problem instance.

Results of our experimental study are shown in tables 1, this contains a comparison to IRT, MTHM, Mulknap techniques and IBM ILOG CPLEX solver on a data-set from literature [16,17], It is clear that the method attendant gives a result better than Mulknap and CPLEX solver either as a solution or as a time over the local search method for MKP

1http://www.diku.dk/Pisinger/codes.html

2http://www.or.deis.unibo.it/staff-pages/Martello/cvitae.html improves the results obtained by the IRT technique with a time greater than the time of the latter method, although it remains our proposal novella usable in real time because time does not exceed one second.

Table 1. *Results on Uncorrelated,strongly correlated and multiple subset-sum Instances compared to IRT ,MTHM , Mumknap techniques and ibm ilog cplex solver. Time columns show the time in seconds.*

Uncorrelated instances										
	MTHM		**IRT**		**Mulknap**		**Cplex**		**Local Search**	
	Z	***Time***	***Z***	***Time***	***Z***	***Time***	***Z***	***Time***	***Z***	***Time***
S 1	806906	0.004	830575	0.293	843374	180	843509	180	842322	0.571
S 2	778781	0.005	791931	0.372	801497	180	802111	180	793371	0.838
S 3	723833	0.008	730805	0.276	740210	180	746721	180	751760	0.683
S 4	755329	0.001	769366	0.404	780777	180	785264	180	782453	0.67
Strongly correlated instances										
	MTHM		IRT		Mulknap		Cplex		Local Search	
	Z	*Time*	*Z*	*Time*	*Z*	*Time*	*Z*	*Time*	*Z*	*Time*
S 1	699757	0.001	752146	0.295	745837	180	751391	180	752423	0.388
S 2	681330	0.001	766349	0.42	767114	180	767667	180	767724	0.55
S 3	629253	0.001	710742	0.28	708087	180	710182	180	711614	0.385
S 4	673521	0.001	726563	0.44	722244	180	725484	180	726493	0.61
S 5	711381	0.001	773263	0.401	765050	180	773154	180	773197	0.532
S 6	661103	0.001	738439	0.28	734766	180	738229	180	738307	0.378
S 7	669063	0.001	742562	0.335	742112	180	743073	180	742773	0.449
S 8	704983	0.005	756424	0.233	749965	180	756468	180	756576	0.332
S 9	688309	0.001	753506	0.346	756370	180	756597	180	754527	0.444
S 10	720932	0.004	785382	0.368	783801	180	785191	180	785644	0.476
Multiple subset-sum instances										
	MTHM		IRT		Mulknap		Cplex		Local Search	
	Z	*Time*	*Z*	*Time*	*Z*	*Time*	*Z*	*Time*	*Z*	*Time*
S 1	747026	0.009	750145	0.189	744773	180	749527	180	750145	0.287
S 2	762816	0.008	767355	0.182	764293	180	765203	180	767355	0.27
S 3	707080	0.018	709369	0.18	706549	180	708524	180	709369	0.266
S 4	722512	0.008	725203	0.243	721480	180	724154	180	725203	0.377

6. Conclusion

In this article we described an improvement of IRT technique. The proposed method succeeds to give better results compared to IRT, Mulknap and CPLEX with reasonable.

The future work on this new heuristic approach includes a depth experimental study in large-scale data-sets.

References

[1] M. Lalami,M. Elkihel, D. Baz and V.Boyer, ”A procedure-based heuristic for 0-1 Multiple Knapsack Problems”, International Journal of Mathematics in Operational Research, vol. 4, No. 3, pp. 214-224, 2012.

[2] Y. Laalaoui, “Improved Swap Heuristic for the Multiple Knapsack Problem” IWANN 2013, Part I, LNCS 7902, pp. 547–555, 2013.

[3] S. Martello, P. Toth. “Knapsack problems: algorithms and computer implementations”. J Wiley. 1990.

[4] J. Dréo, A. Petrowski, D. Taillard, P. Siarry"Métaheuristiques pour L'optimisation difficile" ,Eyrolles (Editions), November 2003

[5] G. Ingargiola and J.F. Korsh, "An algorithm for the solution of 0-1 loading problems", Operations Research, 23(6):1110--1119, 1975.

[6] M.S. Hung and J.C. Fisk, "An algorithm for the 0-1 multiple knapsack problem", Naval Research Logistics Quarterly, 571--579, 1978.

[7] S. Martello and P. Toth., "Solution of the zero-one multiple knapsack problem", European Journal of Operational Research, 4, 1980.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[8] S. Martello and P. Toth., "A bound and bound algorithm for the zero-one multiple knapsack problem", Discrete Applied Mathematics, vol. 3, pp. 275--288, 1981.

[9] D. Pisinger,"An exact algorithm for large multiple knapsack problems", European Journal of Operational Research, vol. 114, pp. 528--541, 1999.

[10] A. Fukunaga, R.E Korf, "Bin Completion Algorithms for Multicontainer Packing, Knapsack, and Covering Problems", Journal of Artificial Intelligence Research, vol. 28, pp. 393--429, 2007.

[11] A. Fukunaga, "A branch-and-bound algorithm for hard multiple knapsack problems", Annals of Operations Research, vol. 184, N. 1, pp. 97--119, 2011.

[12] S. Martello and P. Toth., "Heuristic algorithms for the multiple knapsack problem", Computing, vol. 27, pp. 93--112, 1981.

[13] M. Lalami,M. Elkihel, D. Baz and V .Boyer, "A procedure-based heuristic for 0-1 Multiple Knapsack Problems, International Journal of Mathematics in Operational Research, vol. 4, No. 3, pp. 214--224, 2012

[14] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing", Journal of Heuristics, pages 2:5 - 30, 1996.

[15] R. Raidl, "The multiple container packing problem: A genetic algorithm approach with weighted codings", ACM SIGAPP Applied Computing Review, pages 22 - 31, 1999.

[16] A. Fukunaga., "A new grouping genetic algorithm for the multiple knapsack problem", In Proc. IEEE Congress on Evolutionary Computation, pages 2225--2232, 2008.

[17] A. Fukunaga and Satoshi Tazoe, "Combining Multiple Representations in a Genetic Algorithm for the Multiple Knapsack Problem", In Proc of the 11th IEEE Congress on Evolutionary Computation, pages 2423 - 2430, 2009.